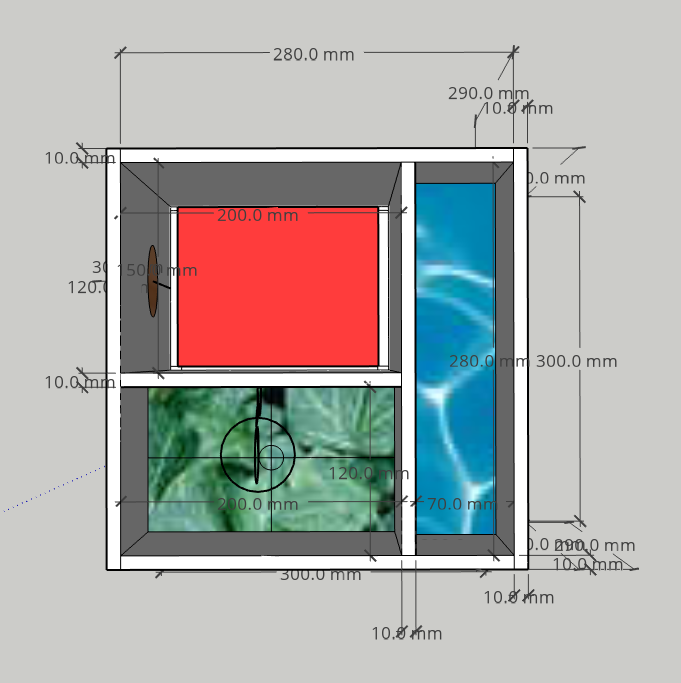
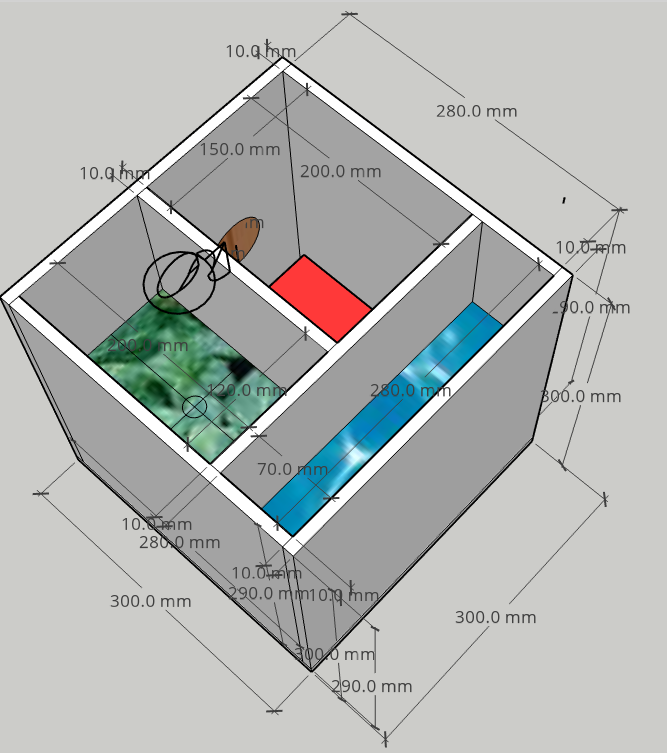
| **유스케이스명** | **유스케이스 개요** | **관련 액터** | **선행 조건** | **테스트 케이스명** | **테스트 케이스 개요** | **테스트케이스 이벤트 흐름** |
| --- | --- | --- | --- | --- | --- | --- |
| App 내에서 식물 설명 열람 | 사용자는 Leave Me Alone Android App을 이용한다. App 내의 식물 설명 페이지를 통해 자신이 기를 식물을 이해한다. | App, 사용자 | 1.사용자는 App을 설치한 상태이다. | 식물 설명 열람 가능 여부 확인 | App 내에서 식물 설명 페이지를 조회한다. | 1. 사용자는 바탕화면 UI-3번 버튼을 눌러서 식물 설명 페이지가 열리는지 확인한다. 2. 식물 설명에 오타나 그림이 빠진지 확인한다. 3. 스크롤이 작동하는지 확인한다. 4. 식물 설명 UI-1번 뒤로가기 버튼을 눌러서 페이지를 빠져나오기가 되는지 확인한다. |
| App 내에서 식물 등록 | 사용자는 App을 이용하며, App 내에 기르고자 하는 식물을 등록한다. | App, 사용자 | 1.사용자는 App을 설치한 상태이다.  2. 사용자의 Phone과 Raspberry Pi는 WiFi로 연결된 상태이다. | App 내에서 식물 등록 가능 여부 확인 | App 내에서 식물 등록을 시도한다. | 1. 바탕화면 UI-4번 버튼을 눌러서 식물 등록 페이지가 열리는지 확인한다. 2. 식물 정보 UI-1번 버튼 (‘+’ 버튼)을 누르면, 식물 등록 페이지로 이동이 되는지 확인한다. 3. 이름, 식물 종류 등이 잘 입력 및 선택되는지 확인한다. 4. 식물 등록 UI-4번, ‘확인’ 버튼이 눌리는지 확인한다. 5. 식물 등록이 잘 됐다는 팝업창이 뜨는지 확인한다. 6. 등록된 정보가 식물 정보 페이지에서 조회가 되는지 확인한다. |
|  |  |  |  | (대안 흐름 테스트) - App내에서 식물 등록 중 WiFi 연결 해제 시 알림 작동 확인 | App에서 식물 등록 중 WiFi 연결 해제 시 이를 알린다. | 1. 식물 등록 페이지에서 WiFi 연결이 해제되면 WiFi 연결을 다시하라는 팝업창을 뜨는지 확인한다. |
|  |  |  |  | (대안 흐름) - App 내에서 데이터 미완성 상태에서 등록 시도. | App에서 식물 등록 중 데이터를 다 입력하지 않고 등록을 시도한다. | 1. ‘이름’이나 식물 종류를 넣지 않고 등록한다.  2. “ ‘이름’을 입력해주세요.” 혹은 “식물 종류를 선택해주세요.” 라는 팝업이 뜨는지 확인한다.  3. 식물이 미등록 되는지 확인한다. |
| App과 Raspberry Pi 간 식물 등록 통신 | 사용자가 App 내에서 식물 등록 시, 이 정보는 Raspberry Pi로 전송된다. | App, Raspberry Pi | 1.App과 Raspberry Pi는 WiFi로 연결된 상태이다. | Raspberry Pi에서 식물 등록 성공 확인 | Raspberry Pi에서 식물 등록 요청을 받았을 때, 관련 정보가 잘 저장되는지 확인해본다. | 1. App이 보낸 식물 등록 요청 시 Raspberry Pi가 이를 수신하는지 확인한다. 표준 출력으로 수신 시 이를 표시한다. 2. Pi측의 ‘plant\_name’이라는 변수에 App에서 보낸 ‘식물 이름’ 데이터가 저장됐는지 표준 출력으로 확인한다. 3. Pi 측의 ‘water\_environ’ 변수에 App에서 보낸 ‘수분 별 식물 종류’ 데이터가 저장됐는지 표준 출력으로 확인한다. 4. Pi 측의 ‘light\_type’ 변수에 App에서 보낸 ‘조명 별 식물 종류’ 데이터가 저장됐는지 표준 출력으로 확인한다. 5. Pi의 ‘registration’ 데이터가 TRUE로 설정됐는지 표준 출력으로 확인한다. |
| Pi 수분 값 초기화 | Raspberry Pi에 식물이 등록되면 그 설정대로 적정 수분 구간을 계산한다. | Raspberry Pi | ‘registration’ 변수가 TRUE이다.  ‘running’ 변수가 FALSE이다. | Raspberry Pi에서 수분 구간 초기화가 되는지 확인. | 식물 등록 시, 적정 수분 구간이 자동으로 계산되어 잘 입력되는지 확인한다. | 1. 최적 수분 구간을 나타내는 데이터가 초기화됐는지 본다. 표준 출력으로 opt\_humid 데이터를 확인한다. 2. 급수 주기 데이터가 초기화됐는지 확인한다. water\_supply\_cycle 데이터 값을 표준 출력으로 확인한다.. 3. 펌프 작동 시간을 나타내는 ‘pump\_time’이라는 데이터를 표준 출력으로 확인한다. 4. ‘water\_set\_on’이라는 수분 데이터 입력 여부를 뜻하는 데이터 공간이 TRUE로 설정되었는지, 표준 출력으로 확인한다. |
| Pi 조명 값 초기화 | Raspberry Pi에 식물이 등록되면 그 설정대로 적정 조명 값을 계산한다. | Raspberry Pi | ‘registration’ 변수가 TRUE이다.  ‘running’ 변수가 FALSE이다. | Raspberry Pi에서 조명 값 초기화가 되는지 확인 | 식물 등록 시, 적정 조명 값이 계산되어 입력되는지 확인한다. | 1. light\_type 데이터 공간의 값에 따라, lux\_min이라는 데이터 공간 값이 초기화되는지 확인한다. lux\_min과 light\_type을 표준 출력으로 확인한다. 2. chlorophyll\_b 라는 엽록소 함량을 나타내는 데이터 공간이 초기화되는지 확인한다. light\_type과 chlorophyll\_b를 표준 출력으로 확인한다. 3. lighting\_type이라는 일광 시간을 나타내는 데이터 공간이 초기화 되는지 확인한다. 이를 표준 출력으로 확인한다. 4. 전구가 켜졌는지 나타내는 On 값이 FALSE로 초기화되는지 표준 출력으로 확인한다. 5. light\_set\_on이라는 조명 관리치 입력 여부를 나타내는 데이터 공간이 TRUE로 설정됐는지 확인한다. 표준 출력으로 이를 확인한다. |
| Pi 자동 관리 시작 | Raspberry Pi는 설정 값이 모두 입력되면 자동 관리를 시작한다. | Raspberry Pi | ‘registration’ 변수가 TRUE이다.  ‘running’ 변수가 FALSE이다.  ‘water\_set\_on’이 TRUE이다.  ‘ligit\_set\_on’이 TRUE이다.  Raspberry Pi가 Hue 전구와 연결 된 상태이다. | 식물 등록 시 Pi가 자동관리를 수행 확인. | Pi에 식물이 등록되면 자동관리를 수행하는지 확인한다. 그리고 관련 부품이 실제 동작하는지 확인한다. | 1. 센싱이 정상적으로 되는지 확인한다. sensed\_data 객체가 값을 표준 출력으로 확인한다. 2. 자동 관리를 여부를 나타내는 running 변수가 TRUE로 설정됐는지 표준 출력으로 확인한다. 3. 수분 센서를 뽑아내면, 측정 값이 낮아지는지 표준 출력으로 확인한다. 또한 관련하여 워터펌프가 작동하는지 확인한다. 4. 조도 센서를 가리면, 측정 값이 낮아지는지 표준 출력으로 확인한다. 또한 관련하여, 전구가 켜지는지 확인한다. |
|
| Pi 급수 필요 여부 확인 | Raspberry Pi는 펌프를 켤 필요가 있는지 상시 확인한다. | Raspberry Pi | ‘running 변수가 TRUE이다. | Pi 급수 필요 기능 동작 여부 확인 | Raspberry Pi가 수분이 적정할 때 급수 필요성을 잘 파악하고 있는지 확인한다. | 1. 물 줄 시간을 지금 당장으로 바꾼다. 2. 펌프 작동이 필요하다고, Check\_pump 함수가 TRUE 값을 반환하는지 확인한다. 3. 관련하여 펌프가 작동하는지 확인한다. |
|  |  |  |  | (대안흐름) - 수분 과다 시, Pi 급수 필요 확인 기능 동작 여부 확인 | Raspberry Pi가 수분이 과다할 때, 급수 필요성을 잘 파악하는지 확인한다. | 1. 수분 센서를 물에 담군다. 2. 물 줄 시간을 현재 시각으로 바꾼다. 3. Check\_pump 함수가 물 주는 주기를 미루고, FALSE를 반환하는지 표준 출력으로 확인한다. 4. 펌프가 미작동하는지 확인한다. |
|  |  |  |  | (대안흐름) - 수분 부족 시, Pi 급수 필요 확인 기능 동작 여부 확인 | Raspberry Pi가 수분이 부족할 때, 급수 필요성을 잘 파악하는지 확인한다. | 1. 수분 센서를 화분에서 뺀 후, 수건으로 닦는다. 2. 물 줄 시간을 현재 시각으로 바꾼다. 3. Check\_pump 함수가 펌프 작동이 필요하다고 TRUE를 반환하는지 표준 출력으로 확인한다. 4. 펌프가 작동하는지 본다. |
|  |  |  |  | (대안흐름) - 적정 수분 구간이며, 급수 시간 아닐 때, Pi 급수 필요 확인 기능 동작 여부 확인 | Rspberry Pi가 적정 수분 구간이지만, 급수 시간이 아닐 때, 급수가 필요 없다가 판단하는지 확인한다. | 1. 적정 수분 구간을 확장시킨다. 2. 급수 시간을 현재 시간과 다르게 정한다. 3. Check\_pump가 펌프 작동이 필요 없다고 FALSE를 반환하는지 확인한다. 4. 펌프가 미작동하는지 확인한다. |
| Pi 조명 필요 여부 확인 | Raspberry Pi는 조명을 켤 필요가 있는지 상시 확인한다. | Raspberry Pi | ‘running 변수가 TRUE이다. | Pi 조명 필요 기능 동작 여부 확인 | Raspberry Pi가 빛이 부족할 때, 조명 필요성을 잘 판단하는지 확인한다. | 1. 최소 광량을 최대치로 설정한다. 2. 조도 센서를 손으로 가린다. 3. 전구를 켜야 한다고 check\_lighting()함수가 TRUE를 반환하는지 표준 출력으로 확인한다. 4. 전구가 켜지는지 확인한다. |
|  |  |  |  | (대안흐름)Pi 조명 필요 기능 동작 여부 확인 - 최소 광랑 충족 | 조명이 충분할 때, 조명 필요성을 잘 판단하는지 확인한다. | 1. 최소 광량을 0으로 설정한다. 2. 전구를 켤 필요 없다고, check\_lighting() 함수가 FALSE를 반환하는지 표준 출력으로 확인한다. 3. 전구가 꺼지는지 확인한다. |
|  |  |  |  | (대안흐름) Pi 조명 필요 기능 동작 여부 확인 - 조명시간 아님 | 조명 시간이 아닐 때, 조명 필요성을 잘 판단하는지 확인한다. | 1. 조명 시간을 현재 시각과 다르게 설정한다. 2. 전구를 켤 필요 없다고 check\_lighting() 함수가 FALSE를 반환하는 지 표준 출력으로 확인한다. 3. 전구가 미작동하는지 확인한다. |
| App 내에서 식물 삭제 | 사용자는 App 내에서 기르던 하는 식물을 삭제한다. | App, 사용자 | 사용자의 smartphone과 Raspberry Pi는 WiFi로 연결 된 상태이다. | App 내 식물 삭제 확인 | 사용자가 App 내에서 식물을 삭제할 수 있는지 확인한다. | 1. 바탕화면에서 식물 정보(바탕화면 UI 5번)를 선택하여 이동이 가능한지 확인한다. 2. 식물 정보 1번 UI, 삭제 버튼이 눌리는지 확인한다. 3. 식물 이름을 가리키며 “‘~~’ 삭제하시겠습니까?” 라는 팝업창이 뜨는지 확인한다. 4. ‘확인’을 버튼이 눌리는지 확인한다. 5. App에서 해당 정보가 삭제되는지 확인한다. |
|  |  |  |  | (대안흐름) App 내 식물 삭제 확인 - 삭제 취소 | 사용자가 App내에서 식물 삭제를 취소할 수 있는지 확인한다. | 1. 사용자가 식물 삭제 확인 팝업 창에서 ‘취소’ 버튼을 누른다. 2. 기존 식물 정보가 유지됐는지 확인한다. |
| App과 Raspberry Pi 간 식물 삭제 통신 | 사용자가 App 내에서 식물 삭제 시, 이 정보는 Raspberry Pi로 바로 전송된다. | App, Raspberry Pi | 1.App과 Raspberry Pi는 WiFi로 연결 된 상태이다. | App과 Raspberry Pi 삭제 통신 확인 | 사용자가 App내에서 식물 삭제 시에 그 정보가 Raspberry Pi에서 삭제 요청을 잘 받는지 확인한다. | 1. App에서 식물 삭제를 수행한다. 2. Pi 측의 ‘plant\_name’이라는 데이터 공간이 “no\_name”으로 설정됐는지 확인한다. 표준 출력으로 이를 확인한다. 3. Pi의 ‘water\_environ’ 데이터 공간이 “nowhere”로 초기화 됐는지 확인한다. 표준 출력으로 이를 확인한다. 4. Pi의 ‘registration’ 변수를 FALSE로 설정됐는지 확인한다. 이는 이제 식물이 삭제됐다는 뜻이다. 표준 출력으로 해당 데이터를 출력해본다. |
| Pi 수분 값 제거 | Raspberry Pi에 식물이 삭제되면 적정 수분 구간이 없다고 바꾼다. | Raspberry Pi | ‘registration’ 변수가 FALSE이다.  ‘running’ 변수가 TRUE이다. | Raspberry Pi 내 수분 정보 삭제 확인. | Raspberry Pi가 식물 삭제를 요청 받으면, Pi 내의 적정 수분 구간을 없애는지 확인한다. | 1. ‘opt\_humid’라는 최적 수분 구간을 나타내는 데이터가 -1~-1으로 설정되는지 확인한다. 해당 데이터를 표준 출력으로 출력해본다. 2. 급수 주기를 초기화 됐는지 확인한다. ‘water\_supply\_cycle’이 -1일로 설정됐는지 표준 출력으로 확인한다. 3. 워터 펌프 작동 시간이 없어졌는지 확인한다. pump\_time이 0 혹은 무효 값인지 표준 출력으로 확인한다. 4. ‘water\_set\_on’이라는 변수가 FALSE인지 확인한다. 이를 표준 출력으로 확인한다. 이는 수분 관리치가 없는지 확인하는 거다. |
| Pi 조명 값 제거 | Raspberry Pi에 식물이 삭제되면 적정 광량 구간이 없다고 바꾼다. | Raspberry Pi | ‘registration’ 변수가 FALSE이다.  ‘running’ 변수가 TRUE이다. | Raspberry Pi 내 조명 정보 제거 확인 | Raspberry Pi가 식물 삭제 요청을 받으면, 적정 광량 값을 없애는지 확인한다. | 1. ‘lux\_min’이라는 최소한의 lux 기준을 나타내는 데이터가 -1로 바뀌는지 표준 출력으로 확인한다. 이는 광량 설정이 없음을 뜻한다. 2. ‘chlorophyll\_b’라는 엽록소B 함량을 나타내는 데이터가 NONE으로 바뀌는지 표준 출력을 확인한다. 3. ‘lighting\_time’이라는 조명 시간을 나타내는 객체 값이 0으로 바뀌거나 무효화되는지 확인한다. 표준 출력으로 확인한다. 4. ‘light\_set\_on’이라는 변수가 FALSE로 설정 됐는지 본다. 조명 관리 설정이 없는지 확인하는 거다. |
| Pi 자동 관리 중지 | Raspberry Pi는 관리를 중지한다. | Raspberry Pi | ‘registration’ 변수가 FALSE이다.  ‘running’ 변수가 TRUE이다.  ‘water\_set\_on’이 FALSE이다.  ‘ligit\_set\_on’이 FALSE이다.  Raspberry Pi가 Hue 전구와 연결 된 상태이다. | Raspberry Pi 자동 관리 중지 확인 | Raspberry Pi가 식물을 삭제했을 때, 자동관리를 중지하는지 확인한다. | 1. App에서 식물을 삭제한다. 2. Raspberry에서 관련 정보가 삭제된다. 3. 센서 측정이 멈추는지 확인한다. sensed\_data 값이 어떻게 바꼈는지 출력해 본다. 그리고 App 쪽에서 받아보는 센서 값도 바뀌는지도 확인한다. 4. 펌프 작동이 멈추는지 확인한다. 사전에 수분 센서를 잘 닦아 허공에 둔다, 그로 인해 펌프는 작동한다. 식물 삭제 시 펌프 작동이 멈추는지 확인한다. 5. 전구가 꺼지는지 확인한다. 사전에 조도 센서를 가려서 조명을 작동 시킨 상태에서, 식물 삭제 시 조명이 꺼지는지 확인한다. 6. running 변수가 FALSE로 바뀌는지 표준 출력으로 확인한다. 이제 Raspberry Pi는 식물 관리를 안하는지 나타내는 값이다. |
| App과 Raspberry Pi 연결 | 사용자는 App에서 WiFi로 휴대폰과 Raspberry Pi를 연결한다. | App, 사용자 | Raspberry Pi가 AP로 기능 중이다. | App에서 Raspberry Pi 연결 | App에서 통신 페이지를 이용해 App과 Raspberry Pi를 연결한다. | 1. 바탕화면에서 통신(바탕화면 UI 5번)을 선택하여 이동한다. 2. ‘App-Rapberry Pi 연결’(연결 UI 1번)버튼을 누른다. 3. 휴대폰의 WiFi 네트워크 탐색창으로 연결되는지 확인한다. 4. Raspberry Pi가 ‘LeaveMeAlone’이라는 채널명으로 검색되는지 확인한다. 5. 해당 네트워크에 접속한다. 6. 접속 후 연결 설정 페이지로 돌아온다. 7. UI 1번 버튼이 푸른색으로 바뀌었는지 확인한다. 또한 UI의 연결 상태를 표현하는 모양도 바꼈는지 본다. |
| App을 이용해 Raspberry Pi와 Hue 전구 연결 | 사용자는 App을 이용해서 Raspberry Pi와Hue 조명 간 연결을 명령한다. | App, 사용자 | Leave Me Alone Raspberry Pi와 Leave Me Alone App이 이미 WiFi로 연결 된 상태이다. | App에서 Raspberry Pi와 Hue 전구 연결 확인 | App 상에서 Raspberry Pi와 Hue 전구를 연결할 수 있는지 확인한다. | 1. 바탕화면에서 통신(바탕화면 UI 5번)을 선택하여 이동한다. 2. ‘Raspberry Pi - Hue Bulb Bluetooth’(연결 UI 2번)버튼이 눌리는지 확인한다. 3. 탐색창이 뜨는지 확인한다. 4. Raspberry Pi에서 검색되는 Hue 전구가 목록에 보이는지 확인한다. 5. 원하는 Hue 전구를 선택하고 ‘확인’ 버튼이 눌리는지 눌러본다. 6. 연결이 완료되면 “Raspberry Pi와 XX전구가 연결 되어있습니다.” 라는 팝업창이 뜨는지 확인한다. 7. 연결이 완료 되었는지 확인한다. 설정을 조정한 뒤 전구가 작동하는지 확인한다. |
|  |  |  |  | (대안흐름) Raspberry Pi와 Hue 전구 연결 중, App과 Raspberry Pi 연결 해제 | Pi와 전구간 연결 시도 중 App과 Pi간 연결이 끊기면, 오류창이 띄워지는지 확인한다. | 1. Pi와 전구 연결 설정 중에 사용자 휴대폰의 WiFi 연결을 차단한다. 2. App에서 WiFi 연결이 해제되었다는 오류 문구를 띄우는지 확인한다. |
| Raspberry Pi에서 Hue 전구 탐색 | Raspberry Pi에서 보이는 전구 목록을 조회한다. | Raspberry Pi | Raspberry Pi는 App에게서 Hue 전구 탐색을 요청받은 상태이다.  App과 통신 가능한 상태이다. | Raspberry Pi에서 전구 탐색 가능 확인 | Raspberry Pi에서 블루투스 기능이 켜지는지 그리고 기기 목록에서 원하는 전구를 선택해 연결할 수 있는지 확인한다. | 1. Pi의 블루투스 기능이 켜지는지 확인한다. 관련 동작 수행 후 Bluetooth 기능이 켜졌는지, 표준 출력으로 확인한다. 2. Search\_bulb()라는 함수를 수행해서 주변의 전구 목록을 알아낼 수 있나 확인한다. 탐색한 전구 목록을 표준 출력으로 확인한다. 3. App에게 탐색한 목록을 보낼 수 있는지 확인한다. App 측에서 갱신된 기기 목록이 보이는지 확인한다. |
|
| Raspberry Pi에서 Hue 전구 연결 | Raspberry Pi에서 특정 전구와 연결한다. | Raspberry Pi | Raspberry Pi는 App에게서 특정 Hue 전구 연결을 요청받은 상태이다.  App과 통신 가능한 상태이다. | Raspberry Pi에서 전구 연결 | Raspberry Pi에서 특정 전구에 대한 연결을 행한다. | 1. App에게서 특정 전구를 지목 받을 수 있는지 확인한다. App에서 받은 전구 ID가 정상인지 표준 출력으로 확인한다. 2. connect\_bulb()라는 함수를 수행하여, 해당 전구와 연결할 수 있는지 확인한다. 표준 출력으로 관련 동작 수행 후 완료 문구가 뜨도록 유도하여 연결을 확인한다. 3. connection\_done() 함수를 수행하여, 해당 전구와 연결이 완료되었다고 App에게 회신할 수 있는지 확인한다. App 측에서 탐색창을 통해 기기 연결 수행이 가능한지 확인한다. |
|
| App 내에서 습도 설정 변경 | App에서 시스템의 습도 설정을 변경한다. | App, 사용자 | Raspberry Pi와 App이 WiFi로 연결 된 상태이다.  식물은 이미 등록 된 상태이다. | App 내에서 습도 설정 변경 확인 | App 내에서 습도 설정을 바꿀 수 있는지 행해본다. | 1. 바탕화면에서 수분관리(바탕화면 UI 1번)를 선택하여 이동한다. 2. 수분관리UI 1번, 적정 토양 습도 슬라이더가 움직이는지 확인한다. 3. 수분관리UI 4번, 설정 저장 버튼이 눌리는지 확인한다. 4. 새 설정을 적용하겠냐는 팝업 창이 뜨는지 확인한다. 5. 팝업창의 ‘확인’ 버튼이 눌리는지 확인한다. 6. 저장 후 해당 내역이 GUI에 유지되는지 확인한다. |
| App 내에서 펌프 작동 시간 변경 | App에서 시스템의 펌프 작동 시간을 변경한다. | App, 사용자 | Raspberry Pi와 App이 WiFi로 연결 된 상태이다.  식물은 이미 등록 된 상태이다. | App 내에서 펌프 작동 시간 변경 확인 | App 내에서 펌프 작동 설정을 바꿀 수 있는지 해본다. | 1. 바탕화면에서 수분관리 (바탕화면 UI 1번)을 선택하여 이동한다. 2. 수분관리 UI 2번, 펌프 작동 시간 버튼이 눌리는지 확인한다. 3. 시간 설정을 다루는 팝업창이 뜨는지 확인한다. 4. 새 시간을 정할 수 있는지 확인한다. 5. 팝업창의 ’확인’ 버튼이 눌리는지 시도한다. 6. 수분 관리 UI 4번, 설정 저장 버튼이 눌리는지 확인한다. 7. 새 설정을 적용하겠냐는 팝업 창이 뜨는지 확인한다. 8. 팝업 창의’ 확인’ 버튼이 눌리는지 확인한다. 9. 새 펌프 작동 시간이 GUI 상으로 표시되는지 확인한다. |
| App 내에서 워터 펌프 작동 테스트 | App내에서 워터 펌프 작동을 테스트 한다. | App, 사용자 | Raspberry Pi와 App이 WiFi로 연결 된 상태이다.  식물은 이미 등록 된 상태이다. | App에서 워터펌프 작동 테스트 확인 | App에서 워터펌프 작동 테스트를 명령할 수 있는지 확인한다. | 1. 바탕화면에서 수분관리(바탕화면 UI 1번)을 선택하여 이동한다. 2. 수분관리 UI 3번, 펌프 작동 테스트 버튼이 누른다. 3. “펌프 작동 테스트가 수행됐습니다.” 라는 팝업창이 뜨는지 확인한다. 4. 실제로 워터펌프가 작동하는지 확인한다. |
| Raspberry Pi가 새 습도 설정 적용 | Raspberry Pi에서 새 습도 설정을 사용한다. | Raspberry Pi | Raspberry Pi와 App이 WiFi로 연결 된 상태이다.  식물은 이미 등록 된 상태이다.  App에게서 새 습도 설정을 요청 받는다. | Raspberry Pi 새 습도 설정 적용 확인 | Raspberry Pi가 새 습도 설정을 전달 받으면, 이를 Pi 내에 새로 잘 저장하는지 확인한다. | 1. App에서 최적 수분 구간을 나타내는 데이터를 새로 받으면, 수신 순간을 표준 출력으로 표시한다. 2. 새 값을 ‘new\_opt\_humid’라고 하여 저장하는지 확인한다. 새 값 저장 전후의 new\_opt\_humid 값을 표준 출력으로 확인한다. 3. Raspberry Pi에서 기존에 최적 수분 구간을 나타내던 ‘opt\_humid’의 값을 new\_opt\_humid 의 값으로 바꾸는지 확인한다. 저장 전 opt\_humid 값과 저장 후 opt\_humid 값이 다른지 확인한다. 그리고 바뀐 값이 new\_opt\_humid와 같은지 표준 출력으로 확인한다. |
| Raspberry Pi가 새 펌프 작동 시간 적용 | Raspberry Pi에서 새 펌프 작동 시간을 사용한다. | Raspberry Pi | Raspberry Pi와 App이 WiFi로 연결 된 상태이다.  식물은 이미 등록 된 상태이다.  App에게서 새 펌프 작동 시간을 요청 받는다. | Raspberry Pi 새 펌프 작동 시간 적용 확인 | Raspberry Pi가 새 펌프 설정을 전달 받으면, 이를 Pi 내에 잘 저장하는지 확인한다. | 1. App에서 펌프 작동 시간을 나타내는 데이터를 새로 받으면 해당 순간을 표준 출력으로 나타낸다. 2. 새 펌프 시간을 ‘new\_pump\_time’에 저장하는지 확인한다. 수신 전후의 값을 표준 출력으로 확인한다. 3. Raspberry Pi에서 기존에 최적 수분 구간을 나타내던 ‘pump\_time’ 값을 new\_pump\_time 의 값으로 바꾸는지 확인한다. 새 값 수신 전후의 pump\_time 값이 new\_pump\_time 값에 맞추아 바뀌는지 표준 출력으로 확인한다. |
| Raspberry Pi가 워터 펌프 작동 테스트 | Raspberry Pi에서 워터 펌프를 시험 작동시킨다. | Raspberry Pi | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -App에게서 펌프 테스트 요청을 받는다. | Raspberry Pi 측 워터펌프 작동 테스트 | Raspberry Pi에서 워터펌프 작동 테스트를 수행할 수 있는지 확인한다. | 1. pump\_act\_test()라는 펌프 작동 함수를 호출한다.  1-1. pump\_act\_test가 digitalWrite()함수를 이용해 n초간 GPIO 출력을 내보내는지 확인한다. 실습용 LED를 이용해서 출력을 확인한다.  1-2. LED 대신 펌프 드라이버와 펌프를 연결하여 실습한다. 펌프가 실제로 물을 빨아올리는지 관찰한다. |
| Raspberry Pi가 워터펌프 작동 | Raspberry Pi에서 워터 펌프를 작동시킨다. | Raspberry Pi | -check\_humid() 함수가 펌프 작동이 필요하다는 값을 반환하다. | Raspberry Pi 측 워터펌프 동적 작동 테스트 | Raspberry Pi에서 센서 데이터에 따라 펌프 작동을 달리하는지 확인한다. | 1. pump\_act()라는 펌프 작동 함수를 호출한다.  1-1. 펌프가 작동하는지 확인한다.  1-2. 내부적으로 check\_humid() 함수를 통해 펌프 작동 여부를 확인하는지 알아본다. 센서를 물에 담구어 과습 상태를 조성한다.  1-3. 센서 값이 바뀌면 이에 따라, 펌프 작동이 필요 없다는 결과를 받을 수 있는지 표준 입출력으로 확인한다.  1-4. 결과에 따라 펌프 출력을 조정하는지 확인한다. 결과 값과 동일히 펌프가 꺼지는지 확인한다. |
|  |  |  |  | (대안흐름) 워터펌프 동적 작동 테스트 - 펌프 재 작동 필요 | 장기간 펌프 작동 시 펌프를 멈추는지 확인한다. | 1. 수분 센서를 화분에서 제거한다. 2. 워터펌프가 작동하는지 본다. 3. 워터펌프가 10초 이상 작동 시 작동을 멈추는지 확인한다. |
| App 내에서 점등 광량 변경 | App에서 시스템의 점등 기준을 변경한다. | App, 사용자 | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -Raspberry Pi와 Hue 전구가 연결된 상태이다. | App 내에서 점등 기준 변경 | App 내에서 시스템의 점등 기준을 변경할 수 있는지 확인한다. | 1. 바탕화면에서 조명관리(바탕화면 UI 2번)를 선택하여 이동한다. 2. 조명관리 UI 2번, 점등 광량 슬라이더가 움직이는지 확인한다. 3. 원하는 최소 광량을 정한다. 4. 조명 관리 UI 1번, 설정 저장 버튼이 눌리는지 확인한다. 5. 새 설정을 적용하겠냐는 팝업 창이 뜨면 ‘확인’을 누른다. 6. 새로 저장된 설정이 GUI 상에 적용되는지 확인한다. 7. 새 기준에 따라 전구가 점등되는지 확인한다. |
| App 내에서 조명 시간 설정 | App에서 시스템의 조명 시간을 정한다. | App, 사용자 | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -Raspberry Pi와 Hue 전구가 연결된 상태이다. | App 내에서 조명 시간 설정 변경 확인 | App 내에서 조명 시간을 바꿀 수 있는지 확인한다. | 1. 바탕화면에서 조명관리(바탕화면 UI 2번)를 선택하여 이동한다. 2. 조명관리 UI 3번, 점등 시간 버튼을 누르면 Time Picker 팝업창이 뜨는지 확인한다. 3. 팝업창에서 원하는 시간을 선택 가능한지 확인한다. 4. 조명관리 UI 4번, 소등 시간 버튼을 누르면, Time Picker 팝업창이 뜨는지 확인한다. 5. 팝업창에서 원하는 시간을 선택할 수 있는지 확인한다. 6. 조명관리 UI 1번, 설정 저장 버튼을 누른다. 7. 새 설정을 적용하겠냐는 팝업 창이 뜨는지 확인한다. 8. ‘확인’을 누르면 GUI 상에 해당 설정이 계속 적용되는지 확인한다. 9. 새 조명시간대로 전구가 켜지는지 확인한다. |
| App 내에서 조명 미사용 | App에서 조명을 안쓰기로 정한다. | App, 사용자 | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -Raspberry Pi와 Hue 전구가 연결된 상태이다. | App 내에서 조명 미사용 | App 내에서 조명 미사용 명령을 내릴 수 있는지 확인한다. | 1. 바탕화면에서 조명관리(바탕화면 UI 2번)을 선택하여 이동한다. 2. 조명관리 UI 5번, 조명 미사용 버튼이 눌리는지 확인한다. 3. UI 내의 모든 조명 설정이 표시되지 않는지 확인한다. 4. 조명관리 UI 6번, 설정 저장 버튼을 통해 설정을 저장 가능한지 확인한다. 5. 전구가 실제로 꺼지는지도 확인한다. |
| App 내에서 조명 시간 조회 | App에서 시스템의 조명 시간을 살핀다. | App, 사용자 | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -Raspberry Pi와 Hue 전구가 연결된 상태이다. | App 내에서 조명 시간 조회 가능 여부 확인 | App 내에서 설정된 조명 시간을 확인 할 수 있는지 알아본다. | 1. 바탕화면에서 조명관리 (바탕화면 UI 2번)을 선택하여 이동한다. 2. 조명관리 UI 4번, 설정된 조명시간 UI를 본다. 조명시간이 최근 저장 내용과 일치한지 확인한다. 3. 조명관리 UI 7번, 뒤로가기 버튼을 눌러서 바탕화면으로 되돌아갈 수 있는지 확인한다. |
| App 내에서 엽록소B 비율 설정 | App에서 엽록소B에 대한 비율을 바꾼다. | App, 사용자 | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -Raspberry Pi와 Hue 전구가 연결된 상태이다. | App 내에서 엽록소 B 비율 설정 변경 확인 | App 내에서 엽록소 B 비율 설정을 변경할 수 있는지 확인한다. | 1. 바탕화면에서 조명관리 (바탕화면 UI 2번) 을 선택하여 이동한다. 2. 조명관리 UI 5번, 엽록소 B 함량에서 원하는 양을 고를 수 있는지 확인한다. 3. 조명관리 UI 6번, 설정 저장 버튼을 누른다. 4. 새 설정을 적용하겠냐는 팝업 창이 뜨는지 확인한다. 5. ‘확인’을 누르면, 실제로 전구의 색조가 바뀌는지 확인한다. |
| Raspberry Pi가 새 광량 설정 적용 | Raspberry Pi에서 새 광량 설정을 사용한다. | Raspberry Pi | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -App에게서 새 광량 설정을 요청 받는다. | Raspberry Pi가 요청 받은 새 광량 설정을 적용할 수 있는지 확인한다. | Raspberry Pi가 App에서 새 광량 설정을 받으면, 그 값대로, 기존 광량 설정을 대체하는지 확인한다. | 1. App에서 최소 광량을 나타내는 데이터를 새로 받으면 표준 출력으로 이를 표시한다. 2. 이를 ‘new\_lux\_min’라고 하여 저장하는지 뒤, 저장 여부를 표준 출력으로 확인한다. 3. Raspberry Pi에서 기존에 최소 광량을 나타내던 ‘lux\_min’ 의 값을 new\_lux\_min 것으로 바꾸는지 표준 출력으로 확인한다. 4. 새 광량 설정대로 전구가 점등 되는지 확인한다. |
| Raspberry Pi가 새 조명 시간 적용 | Raspberry Pi에서 새 조명 시간을 사용한다. | Raspberry Pi | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -App에게서 새 조명 시간을 요청 받는다. | Raspberry Pi 새 조명 시간 적용 여부 확인 | Raspberry Pi가 App에서 새 조명 시간을 요청 받으면 그대로 조명 시간을 갱신하여 사용하는지 확인한다. | 1. App에서 조명 시간을 나타내는 데이터를 새로 받으면 표준 출력으로 이를 표시한다. 2. 이를 ‘new\_lighting\_time’이라 저장하는지 표준 출력으로 확인한다. 3. Rasbperry Pi에서 기존에 조명 시간을 나타내던 “lighting\_time” 객체의 값을 new\_lighting\_time 것으로 바꾸는지 표준 출력으로 확인한다. 4. 새 조명 시간대로 전구가 작동하는지 확인한다. |
| Raspberry Pi가 새 엽록소 B 함량 적용 | Raspberry Pi에서 새 엽록소 B 함량 값을 사용한다. | Raspberry Pi | -Raspberry Pi와 App이 WiFi로 연결된 상태이다.  -식물을 이미 등록된 상태이다.  -App에게서 새 조명 시간을 요청 받는다. | Raspberry Pi 새 엽록소 B 함량 적용 확인 | Raspberry Pi가 App에서 새 엽록소 B 함량을 요청 받으면, 이로 데이터를 갱신하여 쓰는지 확인한다. | 1. App에서 엽록소 B 함량을 나타내는 데이터를 새로 받으면 표준 출력으로 이를 표시한다. 2. 이를 ‘new\_chlorophyll\_b’이라 저장하는지 표준 출력으로 확인한다. 3. Raspberry Pi에서 기존에 조명 시간을 나타내던 ‘chlorophyll\_b’ 객체의 값을 new\_chlorophyll\_b 것으로 바꾸는지 표준 출력으로 확인한다. 4. 새 엽록소 설정대로 조명 색이 바뀌는지 확인한다. |
| Raspberry Pi가 조명 작동 | Raspberry Pi에서 조명을 켠다. | Raspberry Pi | -check\_light() 함수가 조명을 켜야한다는 값을 반환하다.  -전구가 켜진 걸 뜻하는 on 값이 FALSE다. | Raspberry Pi 조명 작동 여부 확인 | Raspberry Pi가 필요할 때 조명을 적절히 켜서 쓰는지 확인한다. | 1. light\_act()라는 조명 작동 함수를 호출하는지 표준 출력으로 확인한다. 2. 밝기를 뜻하는 brightness을 1로 설정하는지 확인한다. 3. 전구 작동을 뜻하는 on 값을 TRUE로 설정하는지 확인한다. 4. 전구가 실제로 켜지는지 확인한다. |
|  |  |  |  | (대안흐름) Raspberry Pi 조명 작동 여부 확인 - 광량 부족 | Raspberry Pi가 이미 조명을 켰음에도 빛이 부족하면 추가적인 조치를 하는지 확인한다. | 1. check\_light() 함수를 호출하는지 표준 출력으로 확인한다. 2. 전구가 아직 어둡다는 결과를 받으면 표준 출력으로 이를 확인한다. 3. 전구를 더 밝게 켜기 위해 light\_brighter() 함수를 호출하는지 표준 출력으로 확인한다. |
| Raspberry Pi가 조명 밝기 증가 | Raspberry Pi에서 조명을 더 밝게 켠다. | Raspberry Pi | -check\_light() 함수가 조명을 켜야한다는 값을 반환하다.  - 전구가 켜진 걸 뜻하는 on값이 TRUE다. | 전구 밝기 증가 확인 | 센싱 값에 따라 Raspberry Pi가 전구 밝기를 증가시키는지 확인한다. | 1. light\_brighter() 라는 조명 작동 함수를 호출하는지 표준 출력으로 확인한다.  2.밝기를 뜻하는 brightness이 기존에서 +1 되는지 표준 출력으로 확인한다.  3. check\_light() 함수를 반환 값에 따라 빛이 적정하면 함수를 종료하는지 디버거로 확인한다.  4. 전구 밝기가 실제로 증가하는지, 별도의 조도계로 확인한다. |
|  |  |  |  | (대안흐름) 전구 밝기 증가-광량 부족 | 전구 밝기를 이미 한 번 증가 시킨 뒤에도, 밝기가 여전히 어두울 때, 밝기 증가를 재귀적으로 행하는지 확인한다. | 1. 최소 광량을 최대치로 설정한다. 2. light\_brighter() 함수를 호출하는지 표준 출력으로 확인한다. 3. 해당 함수가 brightness를 계속 증가시키는지 표준 출력으로 확인한다. 4. 전구 밝기가 실제로 증가하는지 별도의 조도계로 확인한다. |
|  |  |  |  | (대안흐름) 전구 밝기 증가-광량 최대치 | 전구 밝기가 이미 최대일 때, 밝기 값을 더이상 안 바꾸는지 확인한다. | 1. brightness를 245로 설정한다. 2. light\_brighter() 함수가 brightness 값을 바꾸는지 표준 출력으로 확인한다. 3. 전구에 실제 밝기를 조정하는지 확인한다. |
| Raspberry Pi가 조명 밝기 줄임 | Raspberry Pi에서 조명 밝기를 줄인다. | Raspberry Pi | -check\_light() 함수가 조명이 필요 없다는 FALSE를 반환하다.  -전구가 켜진 걸 뜻하는 on 값이 TRUE다. | 전구 밝기 감소 확인 | 전구 밝기 감소 기능이 정상적으로 작동하는지 확인한다. | 1. light\_darker() 라는 조명 작동 함수를 호출하는지 표준 출력으로 확인한다. 2. brightness 값을 감소하는지 표준 출력으로 확인한다. 3. 실제 전구 밝기가 감소하는지 조도계로 확인한다. |
|  |  |  |  | (대안흐름) 전구 밝기 감소 확인 - 광량  과다 | 전구 밝기 감소 기능을 필요에 따라 재귀적으로 행하는지 확인한다. | 1. check\_light() 함수에게서 전구가 아직 밝다는 결과를 받는지 표준 출력으로 확인한다. 2. 조명 밝기 줄임이 반복되는지 brightness 값을 표준 출력으로 확인한다. 3. brightness가 최소값인 1인 경우 조명을 끄는 llight\_deact() 함수를 호출하는지 표준 출력으로 확인한다. 4. 실제로 전구가 꺼지는지 확인한다. |
|
| Raspberry Pi가 조명 끄기 | Raspberry Pi에서 조명을 끈다. | Raspberry Pi | -check\_light() 함수가 조명이 불필요하다는 값을 반환하다.  -brightness 값이 1이다. | 조명 끄기 확인 | Raspberry Pi에서 조명 끄기를 명령했을 때, 이를 잘 수행하는지 확인한다. | 1. light\_deact()라는 조명 작동 함수를 호출하는지 확인한다. 2. on 값이 FALSE로 설정되는지 표준 출력으로 확인한다. 3. 조명이 꺼지는지 확인한다. |

텍스트, 시계, 스크린샷이(가) 표시된 사진

자동 생성된 설명